



Simultaneous Simulated Annealing-Based Crossover Within a Multi-Agent Model for Solving the Green Share-a-Ride Problem

Elhem Elkout^{1,4} , Housseem Eddine Nouri^{2,4}  and Olfa Belkahla Driss^{3,4} 

¹*Ecole Nationale des Sciences de l'Informatique, University of Manouba, Tunisia*

²*Institut Supérieur d'Informatique et de Gestion de Kairouan, University of Kairouan, Tunisia*

³*Ecole Supérieure de Commerce de Tunis, University of Manouba, Tunisia*

⁴*LARIA UR22ES01, Ecole Nationale des Sciences de l'Informatique, University of Manouba, Tunisia*

Keywords: Green Transport, Green Share-A-Ride Problem, Simulated Annealing, Crossover Operator, Multi-Agent.

Abstract: This research addresses the Green share-a-ride problem (Green-SARP), which is an extension of the share-a-ride problem (SARP) by considering a limited driving range of vehicles in combination with limited refueling infrastructure. The goal of Green-SARP is to remove the possibility of a vehicle running out of fuel during a route by allowing refueling at any alternative fuel station. In this work, we present a new simultaneous Simulated Annealing-based Crossover within a Multi-Agent model (SAC-MA) to solve Green-SARP. In fact, adding to the neighbor operators, the crossover operator is integrated to diversify the search allowing to explore new areas in the search space. Experimental studies are carried out in order to evaluate the performance of our approach, based on new generated data instances, allowing to show its efficiency compared to a Simulated Annealing algorithm (SA).

1 INTRODUCTION


In recent years, numerous studies have focused on the Share-a-Ride Problem (SARP) and its various extensions. First proposed by (Li et al., 2014), SARP involves using taxis to simultaneously transport passengers and parcels, inspired by the well-established Dial-a-Ride Problem (DARP) (Cordeau and Laporte, 2007). While DARP aims to minimize total transportation costs and enhance passenger service, SARP introduces the dual objective of managing both passenger transport and parcel delivery. The key assumption in SARP is that passenger service takes precedence over parcel delivery to ensure high-quality service (Li et al., 2014). Initial solutions to the dynamic SARP were developed using heuristic approaches, including the insertion of unfulfilled requests into existing routes, followed by optimization through neighborhood search.


Subsequent research on SARP has introduced enhancements and variants tailored to real-life urban transportation scenarios. For example, (Nguyen et al., 2015) proposed a hybrid transportation model for


Tokyo, enhancing realism by incorporating additional constraints and a time slack strategy for route planning. This approach utilized the Adaptive Large Neighborhood Search (ALNS) heuristic, which begins with a basic greedy insertion of requests and applies simulated annealing as a local search method.

Several notable variants of SARP have also been proposed. The General Share-a-Ride Problem (G-SARP) (Yu et al., 2018) extends SARP by relaxing certain constraints, such as allowing vehicles to serve multiple requests simultaneously without prioritizing passenger or parcel service. To solve G-SARP, a simulated annealing algorithm and a tabu search were developed to evaluate its performance. Similarly, (Beirigo et al., 2018) introduced the Share-a-Ride with Parcel Lockers Problem (SARPLP), focusing on shared autonomous vehicles equipped with flexible compartments for passengers and parcels.

Further developments include a time-dependent model for SARP (Do et al., 2017), incorporating speed windows to reflect realistic urban conditions. Another extension, the cooperative SARP (coop-SARP) (Cavagnini and Morandi, 2021), examines collaboration between different service providers. Additionally, (Yu et al., 2021) introduced the Share-a-Ride Problem with Flexible Compartments (SARPFC), which allows the passenger compartment

^a  <https://orcid.org/0000-0001-7422-7951>

^b  <https://orcid.org/0000-0003-0901-1278>

^c  <https://orcid.org/0000-0003-3077-6240>

to be used for parcel storage, aiming to maximize revenue.

Building on these advancements, the Green Share-a-Ride Problem (Green-SARP) was proposed by (Elkout and Belkahla, 2022) to address environmental concerns. Green-SARP combines the principles of SARP with those of the Green Vehicle Routing Problem (GVRP), incorporating Alternative Fuel Vehicles (AFVs) and the need for refueling at Alternative Fuel Stations (AFS). This formulation aims to reduce the environmental impact of shared mobility while maintaining operational efficiency.

In 2023, (Elkout et al., 2023) introduced an enhanced Simulated Annealing algorithm with a Correction Mechanism (SA-CM) to solve Green-SARP. Their experimental results demonstrated that SA-CM could produce high-quality solutions close to the optimal results obtained by CPLEX, outperforming CPLEX for instances with more than 10 requests. These findings highlight the potential of heuristic-based methods to solve complex urban transportation problems efficiently.

This innovative problem Green-SARP was first introduced by (Elkout and Belkahla, 2022), by leveraging the flexibility of AFVs and integrating AFS nodes into the route design, the Green-SARP ensures the sustainability of urban transport systems while meeting the demands of modern shared mobility.

In this paper, we introduce a simultaneous Simulated Annealing-based Crossover within a Multi-Agent model (SAC-MA) to solve Green-SARP. We conduct computational experiments to assess the performance of our approach, utilizing newly modified data instances, and demonstrate its efficiency compared to a Simulated Annealing algorithm (SA).

2 SIMULTANEOUS SIMULATED ANNEALING-BASED CROSSOVER WITHIN A MULTI-AGENT MODEL

Simulated Annealing is a probabilistic meta-heuristic algorithm, it is based on a natural technique that simulates the cooling of a group of heated atoms using an analogy to thermodynamics, a process known as annealing (Kirkpatrick et al., 1983). Simulated Annealing accepts search movements that temporarily produce degradation of an existing solution to a problem (Kirkpatrick et al., 1983). Simulated Annealing (SA) is a local neighborhood search that requires exploring the search space and accepting solutions with some probabilities (Davtyan and Khcha-

tryan, 2020). An artificial system composed of a population of autonomous agents is called a multi-agent system, in order to achieve their shared goals the agents work cooperatively. Additionally, a multi-agent system is a computational system in which two or more agents cooperate, compete or combine their efforts to accomplish some individual or group objectives (Ferber, 1999). In this work, we propose a new simultaneous Simulated Annealing-based Crossover within a Multi-Agent model (SAC-MA) for solving Green Share-A-Ride Problem. In addition, the fact that Green-SARP is an NP-Hard problem, so the use of a multi-agent system allows distributed and simultaneous processing, which are very complementary. The multi-agent system is composed entirely of interacting agents. Each agent can communicate, coordinate and cooperate with other agents to complete a common aim. It consists of two classes of agents: A master-Agent (MA) and a set of Simulated-annealing Agent (SimA), where the MA agent is the Master of its society and the SimA agents are its Workers/Sub-agents. The figure 1 represents the proposed SAC-MA approach.

2.1 Master Agent

It is the one who interacts with the user, the master-agent (MA) receives the number of Simulated-annealing Agents to create all parameters for the Simulated Annealing-based Crossover algorithm. It is responsible of creating Simulated-annealing Agents (SimA) based on the input number given by the user. Then the MA generates an initial population with different dynamic lists based only on pickup nodes. Noting that the number of dynamic lists depend of the number of the Simulated-annealing Agent given by the user. So, the MA agent provides for each worker-agent its necessary information such as the agent identification (as an autonomous agent and also as a system member), its dynamic list solution from the initial population and the parameters for the Simulated Annealing-based Crossover algorithm. If the stopping criterion is attained, the MA agent chooses the best solution from the last solutions received from Simulated-annealing Agents and displays it as the global solution for the problem.

2.1.1 Solution Representation

The Green-SARP involves a depot, a set of requests, and a set of AFS nodes. The answer to this problem provides 3 travels with 3 AFV each having a 60-gallon fuel capacity and a consumption rate equal to 0.2.

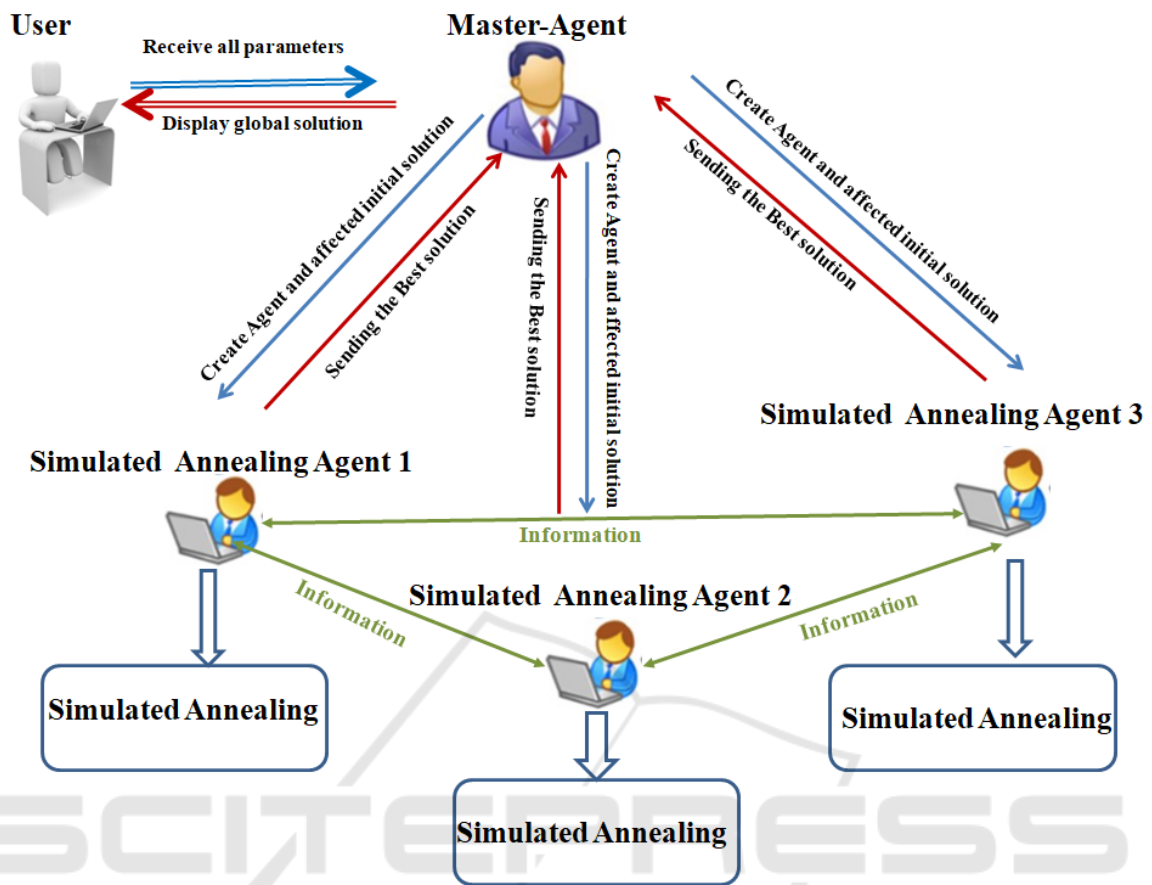


Figure 1: Simultaneous Simulated Annealing-based Crossover within a Multi-Agent Model.

Figure 2 illustrates a sample solution representation and its interpretation into vehicle routes. As shown in Figure 2, the first vehicle is assigned to visit both passengers 1 and 3 and one parcel 7. The vehicle requires more than 60 gallons to serve all affected requests; thus, it decides to visit the AFS node. The remaining vehicles are assigned similarly to how the first vehicle. The solution representation for Green-SARP consists of k dynamic lists. Each list comprises numbers including a depot node, a set of pick-up and drop-off nodes of the passenger and parcel demands, and AFS nodes. The first and the last position in each dynamic list must be respectively an origin and destination depot designate by 0. Each dynamic list represents a vehicle that must serve the different nodes one-by-one from left to right. If the next node is a negative number then the vehicle must visit an AFS node and then continue its travel. To explain the Green-SARP, a sample problem of 8 and 3 vehicles requests is shown in figure 3, in table 3 we present all the necessary parameters:

In Figure 2, the first dynamic list illustrates the route of the first vehicle. It starts at the depot and

Table 1: A simple instance of the Green-SARP.

Parameters	Value
number of requests	$\sigma = 8$
number of parcel requests	$m = 3 \quad \forall p, 0 = \{1, 2, 3, 4, 5\} \quad \forall p, d = \{9, 10, 11, 12, 13\}$
number of passenger requests +	$n = 5 \quad \forall f, 0 = \{6, 7, 8\} \quad \forall f, d = \{14, 15, 16\}$
number of vehicles	$K = 3$
number of AFS	$f = 4$

then visits some pick-up and delivery nodes when the fuel remaining is not sufficient the vehicle visit AFS node denote (-1) then continues its travel and returns to the depot.

2.1.2 Population Initialization

The initial population is generated randomly to increase the diversity and distribution of the individual solutions in the search space. In fact, each new dynamic list solution is based only on pickup nodes and should have a predefined distance from all the other

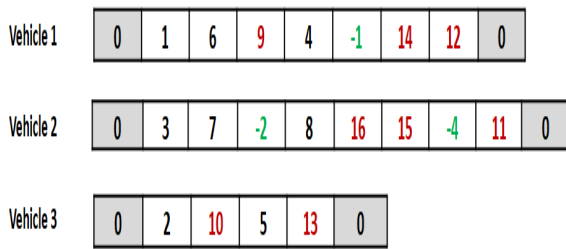


Figure 2: Solution representation for Green-SARP.

dynamic lists to be considered as a new member of the initial population. The dissimilarity distance is calculated by verifying the difference between two dynamic lists in terms of the pickup node order. Noting that a new solution is accepted only if its greater than a fixed threshold of difference equals to 50% percentage.

2.2 Simulated-Annealing Agent

Simulated-annealing Agent (SimA) are created by the Master-Agent, where each one has its own search space area. After generating its initial solution, each SimA agent uses a Simulated Annealing-based Crossover algorithm to explore the search space. To improve the search technique, the SimA agents cooperate and communicate by messages allowing to share their best current solutions among them in order to avoid in each case the reuse of the same research point. SimA agents complete the process by sending their last best solutions to the Master-Agent, which considers the most dominant of them as the global solution for the Green-SARP.

2.2.1 Individual's Solution

The initial solution is created using a random method. **Step 1.** We create an empty list then we add all pickup nodes in random order.

Step 2. We create K empty dynamic lists, we divide randomly the first list for K parts, and each part is affected in one vehicle.

Step 3. we insert after each pickup node her delivery node in each vehicle.

Step 4. We apply the corrector mechanism to correct the route for each vehicle and have a feasible solution, and we insert the origin and the destination depot at first and at the end for each dynamic list. Finally, we insert the AFS nodes consider the tank fuel capacity remaining and we select the nearest AFS node. The figure 3 presents the first three steps for generating the initial solution. After creating the K dynamic lists and applying the correct mechanism in each vehicle, we found a solution we called it "deactivated solution".

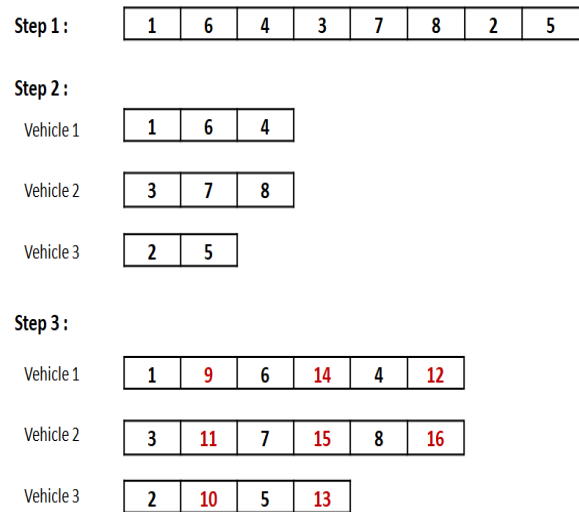


Figure 3: The first steps for generating initial solution for Green-SARP.

Finally, where we insert the depot and the AFS nodes if necessary we find the complete solution named "activated solution". The figure 4 illustrates the initial solution after checking all the constraints.

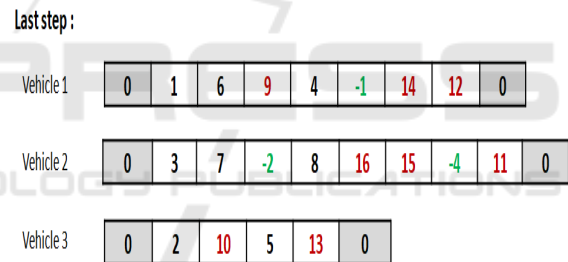


Figure 4: The last step for generating initial solution for Green-SARP.

2.2.2 Neighborhood

The neighborhood move begins with a feasible solution and progresses to the construction of new neighboring solutions of this solution. At each iteration, one of the four neighborhood moves is chosen randomly. In fact, the choice of this neighborhood move is justified by applying the intensification technique to exploit the search space. Each neighborhood move relaunches the search from a current solution to attain an elite solution of the search space. These moves are called: SWAP, Permutation, Insertion, and Reverse. Each Neighborhood moves in the dynamic lists before inserting the AFS nodes and the origin and destination depots, i.e these moves and operators are applied to the "deactivated solution". In the following, we present these four moves.

(1) Permutation. The permutation move is per-

formed by randomly selecting two positions, say p_1 and p_2 ($p_1 \neq p_2$) in the same dynamic list i.e in the same vehicle, and then exchange the two nodes in the two selected positions.

(2) **SWAP.** The SWAP move is performed by randomly selecting many positions, in the same dynamic list and then exchanging the nodes in the positions selected.

(3) **Reverse.** Reversing the sequence of the elements included among two randomly chosen positions p_1 and p_2 ($p_1 \neq p_2$) in the same dynamic list.

(4) **Insertion.** in the first time selecting randomly two positions from the same dynamic list symbolized by p_1 and p_2 ($p_1 \neq p_2$), and the node of the position p_2 is inserted immediately before position p_1 .

2.2.3 Crossover Operator

The crossover operator is necessary for the entire procedure, enabling the creation of new offspring vehicles by combining two parent vehicles in order to reach new promising areas in the search space. To apply this operator, first of all a selection operator is used to randomly select two different parent vehicles V_1 and V_2 . On the other hand, by counting the same length equals exactly to two pickup travels the one point crossover is applied else it is the two point crossover. On the other hand, by having different length, two sub-methods are used, `cutFirstPart` and `cutRemainPart`, to divide the long list of the first parent vehicle into two parts based on the length of the second parent vehicle. Then, the one point crossover is applied if the length of the second vehicle is equal exactly to two pickup travels else it is the two point crossover. Finally, the remaining elements, obtained by the `cutRemainPart` method, are distributed randomly between the two new offspring vehicles.

One Point Crossover: a point on both selected vehicles is fixed randomly, and designated a 'crossover point'. The nodes to the Right of that point are permuted between the two vehicles, which results two offspring. the first child gets the current value of the first vehicle and the second child takes the value of the second vehicle.

Two Point Crossover: two crossover points are generated randomly from the selected vehicles. The pickup nodes between the two points are permuted between these vehicles.

2.2.4 Simulated Annealing-based Crossover

The SA procedure starts by initializing the current temperature T_0 and generating a one random initial solution X , and Sets the current best solution to de-

note X_{best} , and the current best objective function of X denotes $F_{best} = F(X)$. To improve the current solution X , the corrector mechanism is performed to obtain a feasible solution. In each iteration, a new solution Y is generated from the exploitation of the space of search using the neighborhood move based on a random method or from the exploration of search space using the Crossover operator. To select which neighborhood move or crossover operator to use, the algorithm first generated a randomly p value. After that, the objective function values of X and Y are evaluated using the formula, $\Delta = F(Y) - F(X)$. Y is better than X if Δ is positive, in that case Y will replace X . Otherwise, Y will replace X with a probability equal to $\exp(\Delta/T)$. So the accepted solution Y is compared to X_{best} , if $F(Y)$ is better than F_{best} consequently replace F_{best} with $F(Y)$. The current temperature T_0 is decreased after $Iter$ iterations according to the formula $T = \alpha * T_0$. If one of the termination criteria is satisfied, the algorithm will end. There is one termination condition in this algorithm: The final temperature T_f is attained $T_f = T$, this stopping criteria consider the condition of Non-improve: No improvement is reached in succession temperature reductions. Figure 5 shows the flow chart of the proposed SAC heuristic.

3 EXPERIMENTAL STUDIES

To evaluate our model and demonstrate the benefits of using multi-agent systems, we developed an enhanced Simulated Annealing called Improved Simulated Annealing-based Crossover (SAC), which is based on an initial population of solutions. Both the Improved Simulated Annealing (SA) and Simultaneous Simulated Annealing-based Crossover within a Multi-Agent Mode (SAC-MA) were implemented in Java on a PC with Intel processor core i7 vPro and 32 GB of RAM, using the Eclipse IDE to code these approaches and the Jade platform to create the SAC-MA multi-agent system.

3.1 Test Instances

To evaluate and compare the efficiency of these approaches, numerical tests were conducted using the data set generated by Masmoudi et al (Masmoudi et al., 2019) from the literature on the green dial-a-ride problem, and the parameters provided by Li et al. for SARP (Li et al., 2014).

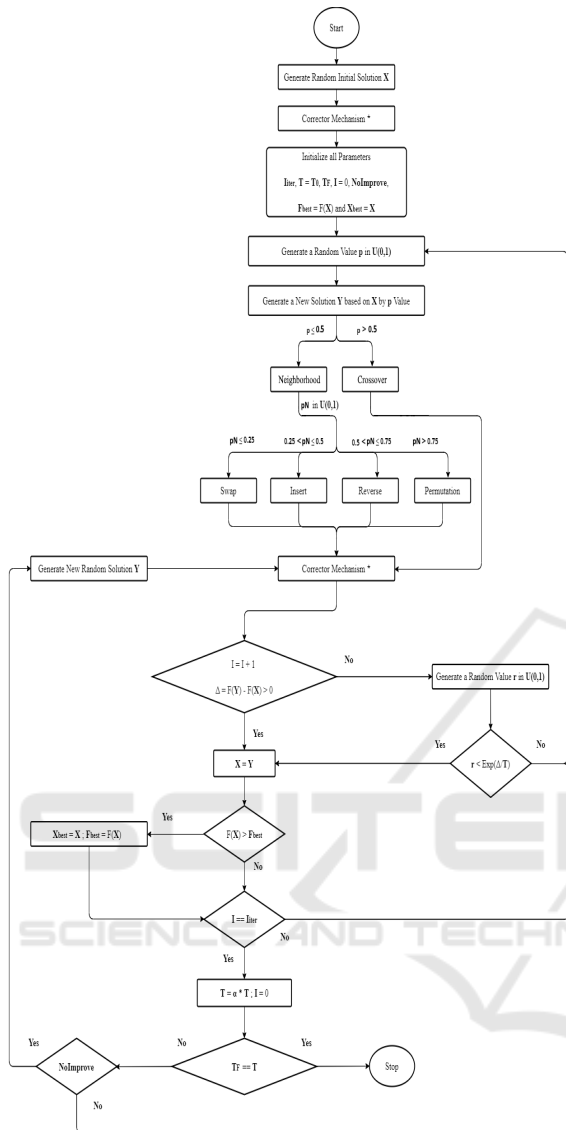


Figure 5: A flow chart of Simulated Annealing-based Crossover.

3.2 Parameter Tuning

The SA metaheuristic uses five parameters: Iter, T0, TF, Non-improving and α .

- Iter: indicates the number of iterations for the search to be proceeded at a specific temperature,
- T0: indicates the initial temperature,
- TF: describes the final temperature when the SA procedure stops,
- Non-improving: is the number of temperature reductions during which the value of the objective function is not improved,

- α : is the factor in charge of the cooling schedule proposed by (Kirkpatrick et al., 1983).

Following an experimental study, the ‘popsize’ parameter was set to 10 agents for small instances, 50 for medium instances (fewer than 40 requests), and 100 for large instances (more than 40 requests). The Non-improving parameter was set to 100 for small instances, 70 for medium instance, and 50 for large instance. The other parameter values are defined as follows:

- Iter = 1000,
- T0 = 12,
- TF = 0,001.

Therefore, we chose for the rest of the parameters the best values used in the literature of SARP variants (Yu et al., 2018), (Yu et al., 2021) $\alpha = 0.9$.

3.3 Computational Results

To evaluate the performance of our SAC-MA approach, we established a rigorous comparative methodology. Our results are compared to those obtained by the SA algorithm developed by (Elkout et al., 2023). Tables 2 present the comparison of SAC-MA with the SA algorithm (Elkout et al., 2023) for small and large instances. They display the optimal value obtained by SA, followed by the best value, the CPU time, and the average value obtained by SAC-MA.

Figures 6 illustrate that SAC-MA performs better and surpasses SA-CM in most cases for both small and large instances. These best results can be explained by the distributed search space in our proposed approach SAC-MA, and the cooperative aspect between the different agents.

Finally, figures 7 shows that SAC-MA is faster in terms of CPU time compared to SA-CM. These excellent results confirm the efficiency of using a multi-agent system, which primarily aims to reduce execution time through the cooperation and communication between Simulated Annealing agents, as well as the specific stopping condition employed in SAC-MA. Indeed, the counter for iterations without improvement (Non-improving) is not incremented solely when an agent finds a locally non-improving solution. The search for the dominant solution by an agent also depends on global solutions. In other words, even solutions found by other agents are considered non-improving if one agent reaches them. This interaction significantly reduces the required CPU time by optimizing convergence towards a high-quality global solution.

Table 2: Comparison of SAC-MA and SA-CM results for small instances.

Instance	Nb-req	Nb-AFS	Nb-Veh	SAC-MA		SA-CM		CPU SAC-MA (minute)	CPU SA-CM (minute)
				Best	AVG	Best	AVG		
A0-10-1	10	3	1	65,742	65,279	64,8	61,2	1,73	8,21
A0-10-1	10	3	2	102,410	102,090	98,7	94,4	4,46	16,08
A0-10-2	10	3	1	54,972	53,605	54	50,9	1,74	8,29
A0-10-2	10	3	2	64,043	63,648	62,1	58,1	4,48	16,35
A0-12-3	12	4	2	149,992	148,422	145	142	5,73	19,92
A0_20_1	20	6	4	523,238	453,204	490,07	406,3	11,23	40,79
A0_20_2	20	6	4	584,415	500,328	545,35	472,3	11,96	42,58
A0_25_4	25	8	5	750,295	652,113	668,33	572,8	13,34	47,32
A0_25_5	25	8	5	605,91	544,503	577,12	540,4	13,46	47,48
A0_30_8	30	9	6	679,382	660,314	653,75	623,8	14,92	51,86
A0_30_9	30	9	5	696,929	650,985	651,87	604,2	14,66	50,92
A0_35_11	35	11	5	765,68	658,534	670,56	635,4	15,33	52,69
A0_35_12	35	11	6	624,543	587,32	615,85	574,3	14,71	51,15
A0_40_14	40	12	7	798,553	764,772	710,52	699,9	15,85	54,21
A0_40_15	40	12	6	790,632	752,48	742,56	704,1	15,49	53,46

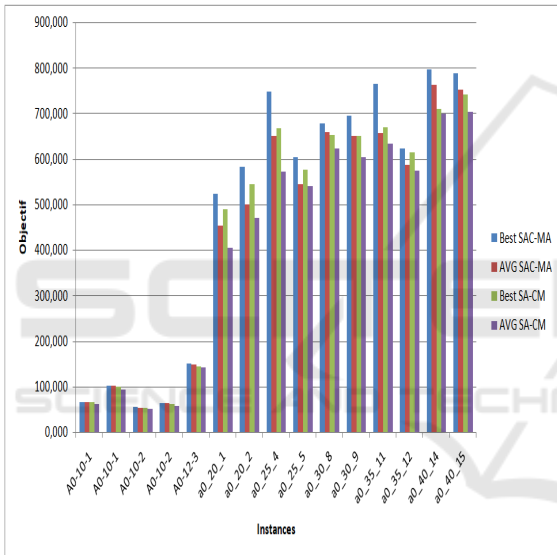


Figure 6: Comparison between SA-CM and the proposed SAC-MA.

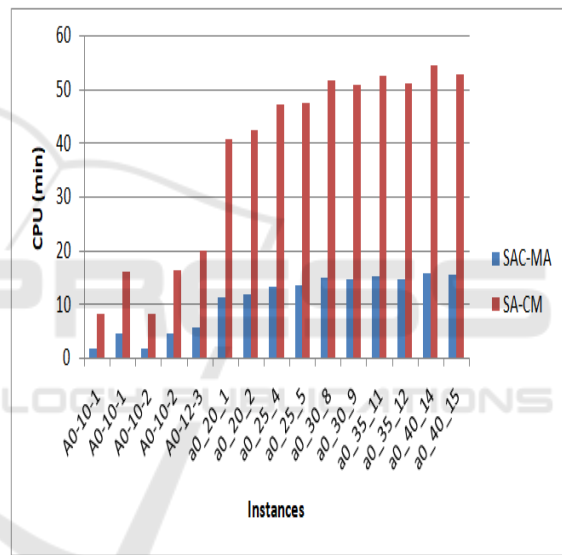


Figure 7: Comparison between SAC-MA and SA-CM in terms of CPU time.

4 CONCLUSIONS

In this study, we introduce a Multi-Agent model called simultaneous Simulated Annealing-based Crossover within (SAC-MA) to solve the Green-SARP. SAC-MA consists of two classes of agents: the Master-Agent which is responsible of managing inputs and outputs of the system and a set of Simulated-annealing Agents allowing to cooperate and communicate between them to increase neighborhood search efficiency and identify promising areas and find the best solution. The experimental results showed that SAC-MA produced the best set of solutions compared to the Simulated Annealing algorithm.

In future work, we plan to adapt our approach to a multi-objective version of the Green-SARP, and compare the results to other approaches from the literature. Furthermore, the proposed SAC-MA algorithm could be used also for future SARP extensions, such as Green general SARP or Green multi-depot SARP and Green-SARP with flexible compartments.

REFERENCES

Beirigo, B., Schulte, F., and Negenborn, R. (2018). Integrating people and freight transportation using shared autonomous vehicles with compartments. *IFAC-Pap.*, 51(9):392–397.

- Cavagnini, R. and Morandi, V. (2021). Implementing horizontal cooperation in public transport and parcel deliveries: The cooperative share-a-ride problem. *Sustainability* 2021, 13(8).
- Cordeau, J.-F. and Laporte, G. (2007). The dial-a-ride problem: Models and algorithms. *Annals of Operations Research*, 153(1):29–46.
- Davtyan, A. and Khachatryan, S. (2020). Simultaneous multi-start simulated annealing for capacitated vehicle routing problem. *WSEAS TRANSACTIONS on COMPUTER RESEARCH*, 8(4598).
- Do, P.-T., Nguyen, N.-Q., and Pham, Q.-D. (2017). A time-dependent model with speed windows for share-a-ride problems: A case study for tokyo transportation. *Data and Knowledge Engineering*, 114:67–85.
- Elkout, E. and Belkahla, D. O. (2022). Modelling and solving the green share-a-ride problem. In *In: Fujita, H., Fournier-Viger, P., Ali, M., Wang, Y. (eds) Advances and Trends in Artificial Intelligence. Theory and Practices in Artificial Intelligence., IEA/AIE 2022*, pages 648–658. Lecture Notes in Computer Science(), vol 13343. Springer, Cham.
- Elkout, E., Nouri, H. E., and Belkahla, D. O. (2023). Simulated annealing for the green share-a-ride problem. In *2023 IEEE International Conference on Artificial Intelligence & Green Energy (ICAIGE), Sousse, Tunisia, 2023*.
- Ferber, J. (1999). Multi-agent systems: An introduction to distributed artificial intelligence addison wesley longman, harlow. *International Journal of Intelligence Science*, 5(4).
- Kirkpatrick, S., Jr, G. C., and MP, V. (1983). Optimizations by simulated annealing. *Science*, 220(4598):671–680.
- Li, B., Krushinsky, D., Reijers, H., and Woensel, T. (2014). The share-a-ride problem: people and parcels sharing taxis. *European Journal of Operational Research*, 238(1):31–40.
- Masmoudi, M., Hosny, M., D., and E: (2019). An adaptive large neighborhood search heuristic for the green dial-a-ride problem. *ISTE Ltd and John Wiley & Sons, Inc. Solving Transport Problems: Towards Green Logistics*.
- Nguyen, N., Dung, N., Do, P., Le, K., Nguyen, M., and Mukai, N. (2015). People and parcels sharing a taxi for tokyo city. *the 6th International Symposium on Information and Communication Technology*.
- Yu, V., PAY, I., AANP, R., and S-W:, L. (2021). Simulated annealing with mutation strategy for the share-a-ride problem with flexible compartments. *Mathematics*, 9(18,).
- Yu, V., Purwanti, S., PerwiraRedi, A., Chung-Cheng, L., Suprayogi, S., and Jewpanya, P. (2018). Simulated annealing heuristic for the general share-a-ride problem. *Eng. Optim*, 50(7,):1178–1197.